**Spring Core and Maven**

**Exercise-1,2&4:**

package com.library.repository;

import java.util.HashMap;

import java.util.Map;

public class BookRepository {

    private final Map<String, String> books = new HashMap<>();

    public BookRepository() {

        // Sample data

        books.put("101", "Spring in Action by Craig Walls");

        books.put("102", "Clean Code by Robert Martin");

    }

    public String getBookInfo(String bookId) {

        return books.getOrDefault(bookId, "Book not found with ID: " + bookId);

    }

}

------------------------

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public String getBookDetails(String bookId) {

        return bookRepository.getBookInfo(bookId);

    }

}

-------------------------------

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean("bookService", BookService.class);

        System.out.println("Book 101: " + bookService.getBookDetails("101"));

        System.out.println("Book 102: " + bookService.getBookDetails("102"));

        System.out.println("Book 999: " + bookService.getBookDetails("999"));

    }

}

-------------------

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source> <!-- Java 1.8 -->

<maven.compiler.target>1.8</maven.compiler.target>

<spring.version>5.3.20</spring.version> <!-- Spring 5.x works with Java 8 -->

</properties>

<dependencies>

<!-- Spring Context (Core) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin (Java 8) -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

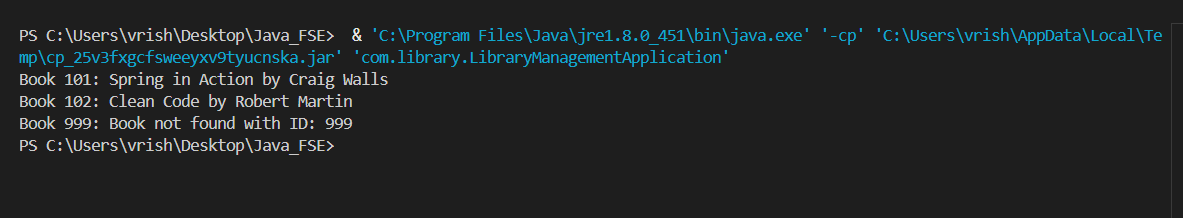
</plugin>

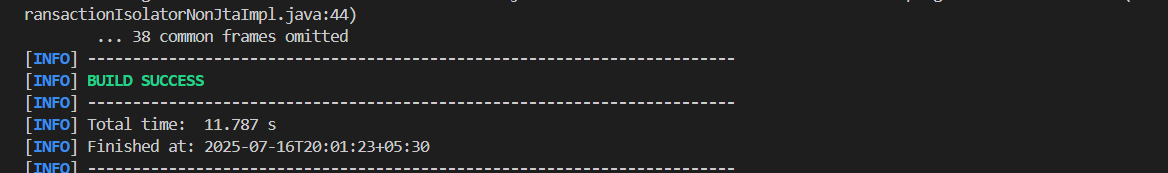
</plugins>

</build>

</project>

**Output:**

****



**Spring Data JPA**

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

    @Id

    @Column(name = "co\_code")

    private String code;

    @Column(name = "co\_name")

    private String name;

    public String getCode() {

        return code;

    }

    public void setCode(String code) {

        this.code = code;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    @Override

    public String toString() {

        return "Country [code=" + code + ", name=" + name + "]";

    }

}

----------------------------

package com.cognizant.ormlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

----------------------------

package com.cognizant.ormlearn.service;

import java.util.Optional;

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

    @Autowired

    private CountryRepository countryRepository;

    @Transactional

    public List<Country> getAllCountries() {

        return countryRepository.findAll();

    }

    @Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

    Optional<Country> result = countryRepository.findById(countryCode);

    if (!result.isPresent()) {

        throw new CountryNotFoundException();

    }

    return result.get();

}

@Transactional

public void addCountry(Country country) {

    countryRepository.save(country);

}

@Transactional

public void updateCountry(String countryCode, String countryName) throws CountryNotFoundException {

    Country country = findCountryByCode(countryCode);

    country.setName(countryName);

    countryRepository.save(country);

}

@Transactional

public void deleteCountry(String countryCode) {

    countryRepository.deleteById(countryCode);

}

}

------------------

package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {

    public CountryNotFoundException() {

        super("Country not found");

    }

}

-------------------

package com.cognizant.ormlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

testGetAllCountries();

LOGGER.info("Inside main");

}

private static void testGetAllCountries() {

LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country:{}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Country not found", e);

}

LOGGER.info("End");

}

}

A screen shot of a computer

AI-generated content may be incorrect.

**Difference Between JPA,Hibernate and Spring Data JPA**

Without offering any real implementation code, the Java Persistence API, or PA, is merely a specification—a collection of guidelines and interfaces that specify how to map Java objects to a relational database. One The most widely used implementation of the JPA specification is Hibernate, an Object-Relational Mapping (ORM) framework that serves as the foundational engine for carrying out the database operations specified by JPA. 2. Data from Spring Conversely, JPA is a higher-level abstraction that is constructed on top of a JPA provider, such as Hibernate, rather than an implementation in and of itself. 3. Its main objective is to drastically cut down on boilerplate code so that developers can create complex data access repositories by just defining an interface. This eliminates the need to manually write implementations for common CRUD